法政大学学術機関リポジトリ
HOSEI UNIVERSITY REPOSITORY

PDF issue: 2025-07-07

同時進行する時系列データからのTransformerによる予測モデルの構築

Hasegawa, Yuta / 長谷川, 雄大

(出版者 / Publisher)
法政大学大学院情報科学研究科
(雑誌名 / Journal or Publication Title)
法政大学大学院紀要. 情報科学研究科編
(巻 / Volume)
17
(開始ページ / Start Page)
1
(終了ページ / End Page)
6
(発行年 / Year)
2022-03-24
(URL)
https://doi.org/10.15002/00025265

同時進行する時系列データからのTransformerによる予測モデルの構築
Building a Forecasting Model with Transformer from Concurrent Time Series Data

長谷川 雄大 *
Yuta Hasegawa
法政大学大学院　情報科学研究科　情報科学専攻
Email: 20t0010@cis.k.hosei.ac.jp

***Abstract*—In this paper, we propose a method to forecast the future of time series data using Transformer. The forecast can be widely utilized in various areas, such as product demand prediction and pandemic analysis of infectious diseases. Many methods have been proposed to analyze the time series data such as ARIMA and neural network. Especially, the Transformers has been attracting researcher's attention to analyze the time series data in recent years.**

In order to precisely forecast the future, a large amount of data is required, but in some cases, such as retail stores, enough data is not always available. Therefore, in this research, we propose a model that combines two types of Transformers ,such as time and space Transformers, to supplement prediction from the relationship between the data. The Time Transformer makes predictions based on the sales performance of the target product, and The Space Transformer infers the relationship between the target product and the others.

In experiments, we showed that the proposed model improved the prediction performance over the ARIMA and the conventional Transformers in the task of predicting the volume of the sales one week ahead.

1. まえがき

近年，ビッグデータの活用方法に注目が集まっている．活用方法の一例として，時系列データの予測がある．時系列データの予測は，活用の範囲が広く，例えば，商品の需要や気温，感染症の日々の感染人数を予測することができる．中でも，需要予測は欠品による販売機会の損失や，過剰な在庫による管理コストの削減につながるため，企業や小売店からの注目も高い．例えば，[1]では, 予測性能が 10〜20% 上昇した結果，在庫管理のコストが 5%減少し，収益が 2〜3%増加したことが報告されている．このように，需要予測は企業，あるいは小売店に及ぼす影響は大きく，注目のタスクになっている．

現実の世界にある多くの時系列データは，データ間に何らかの関係性があり，その関係性を活用することが予測性能の向上につながる．例えば，小売店においては，商品の販売実績は，ある商品 A と B は同時に売れやすいという傾向や，全体の販売量が上がっているため商品 A も売れやすくなった等の関係性があると考えられる．しかし，データの活用が注目され始めたのは最近のことであるため，十分なデータが集まっていない企業や小売店は多い．そこで本稿では，そのような企業や店舗でも短い時系列データから高精度に予測を行うことができるように，データ間の関係から予測を補うことを考えた．具体的には，同時進行している時系列データに対して，時間軸方向の学習を行うモデルと，時系列データ間の関係性を学習するモデルを組み合わせることで，自身の過去の履歴だけでなく，他データとの同時性を考慮し，予測モデルの高精度化を目指した．

2. 関連研究

時系列データに対する予測の手法として，統計的なモデルを用いた手法がある．統計的なモデルでの予測は，過去のデータをもとに，トレンドや季節変動の影響を考慮し，予測を行う．中でも原系列の差分を取って，トレンドの影響を取り除き，過去の自身の値と，過去の誤差から将来の値を予測する ARIMA モデルは，予測問題において大きな功績をあげている [2]．しかし，需要は様々な要因によって変動するため，その要因をすべてモデルに反映させることは困難である．また，予測する対象ごとにモデルを構築する必要があるため，手間がかかるという問題も存在する．

一方，ニューラルネットワークを用いた予測の方法についても研究が進められている．ニューラルネットワークを用いることで，過去データにおける有用な特徴量を自動で学習し，予測に用いることができる．ニューラルネットワークによる時系列データの予測手法として，RNN(Recurent Neural Netwark) がある．RNN はある時点の中間層の出力をその次の時点の入力に加え，中間層に伝えることで，時系列情報を加味し，ニューラルネットワークで予測を行うことができる．近年は，この RNN を改良した LSTM(Long Short-Term Memory) が，時系列データの予測において主流となっている [3]．LSTM の問題点として，逐次的に値を処理していくため，過去の値を直接参照できない点が挙げられる．周期が明確なデータに対しては，直接参照した方が情報の損失がないため，予測性能の向上が期待できる．

再帰や畳み込みなどのコストのかかる計算を行わないモデルとして，自然言語処理の分野で Transformer が登場した．Transformer は，Attention のみを用いて構成されたモデルであり，現在の翻訳タスクはこの Transformer をベースにしたモデルが主流になっている [4]．Attention とは，入力されたデータのどこに着目するべきなのかを計算する機構である．時系列データの予測においても，近年，Transformer を適用する手法に注目が集まっている [5][6]．[5] では，数値を文字列ととらえ，時系列データの予測と機械翻訳のタスクを関連付け，予測を行った．[6] では，Transformer をベースとしたモデルでインフルエンザの感染者数を予測した．2010 年から 2018 年のデータを用いて学習を行い，既存の ARIMA モデルや LSTM モデルよりも予測性能が向上したことを示した．一般に，ニューラルネットワークを用いた予測において，予測の対象となる時系列データと相関のあるデータを特徴量として用いた場合，相関性を学習し，予測に貢献することが期待できる．しかし，Transformer を用いた予測において，特徴量に他の時系列データを複数束ねて用いた場合，予測性能が落ちてしまうことが予備実験で判明した．

* Supervisor: Prof. Satoru Fujita

また，歩行者の経路予測に，Transformer を用いた研究として，[7] がある．この研究では，各歩行者の時間方向についての学習を行う Temporal Transformer と，歩行者間の隣接関係をグラフ構造にし，歩行者間の相互作用を学習する Spatial Transformer を用いることで，従来の時間方向だけの学習ではなく，空間方向についても学習を行った．本稿では，この Spatial Transformer の着想をもとに，時間軸方向の学習と時系列データ間の関係性の学習をそれぞれ別のモデルで行う．これにより，それぞれの学習に特化させることで，他の時系列データがノイズにならず，むしろ予測に貢献できるようなモデルを目指した．

3. Transformer を用いた予測モデル

3.1. 問題設定

本稿で扱う問題について定義する．本稿では，N 種類の時系列データの T 時点までのデータを用いて，その N 種類の時系列データの $T+1$ 時点の値を予測する．すなわち，ある時系列データ i の時刻 t における値を x_t^i とした時，入力 X と出力 Y は以下の式のようになる．

$$X = (x_1^1, x_2^1, ..., x_T^1, x_1^2, x_T^2, ..., x_T^n) \quad (1)$$

$$Y = (x_{T+1}^1, x_{T+1}^2, ..., x_{T+1}^n) \quad (2)$$

X の次元数は，$N \times T$，Y の次元数は N である．

3.2. 提案モデルの概要

本稿で提案するニューラルネットワークのモデルの全体図を図 1 に示す．モデルは最初の処理として，入力データに対して，Embedding Layer で次元を拡張し，位置情報を付与する．Embedding Layer における次元の拡張は 2 段階で行われる．1 段階目は入力データから，3 種類の特徴量を生成し，それぞれ 2 次元に拡張する．2 段階目では，さらにその特徴量を表現するノード数の拡大を行う．そして，Time Transformer で各時系列データごとに，時間軸方向についての学習を行う．その後，時系列方向の学習を終えたデータに対して，Space Transformer で時系列データ間の関係性を学習する．Space Transformer ではデータ数次元と時間次元を入れ替えることで，データ間の関係性の学習を行う．例えば，商品の需要予測の場合，図 2 のように，Time Transformer では各種商品の時間方向について学習を行い，Space Transformer では商品の関係性の学習を行う．これにより，従来時間方向だけであった時系列データの学習をデータ間の関係性の学習に拡張することを可能にする．最後に，Output Layer で予測値を出力する．以下で，各機構の詳細について説明を加える．

3.3. 入力データの極座標への変換

Transformer では，各層の後に特徴量次元で正規化を行う Layer Normalization[8] を行っている．Layer Normalization を行うことで，中間層における入力データの存在範囲が安定し，勾配消失が起きにくくなる．Transformer はもともと自然言語の分野で登場したモデルであり，モデルへの入力は単語の関係をベクトルで表した値を入力している．そのため，特徴量間で正規化を行っても，各特徴量間の関係は維持される．しかし，時系列データの予測においては，観測された値自体に意味があり，特徴量をそのまま実数で表現した場合，特徴量間で正規化を行うとその値の絶対値が消えてしまう．

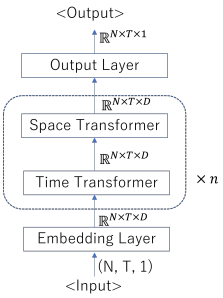


図 1: 提案モデル

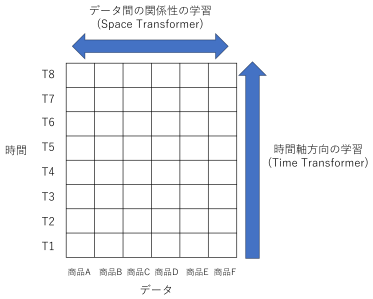


図 2: Time Transformer の学習と Space Transformer の学習

そこで，本稿では，入力に使用する値を絶対値としてシステムに入力するのではなく，単位円上の角度で表すことで 2 次元に拡張し，Layer Normalization が適用可能なデータ表現とした．角度として値を表すことで，Layer Normalization を行って，各ノードの絶対値を正規化しても，その特徴量間の関係を維持することができる．また，本稿では特徴量として，その時刻における値，その時刻と 1 時刻前との差分，週番号を用いた．これらの 3 つは，自身のデータのみで得られる情報であるため，採用した．週番号 w_t については式 (3)，その他の値 x_t は式 (4) を用いて拡張する．

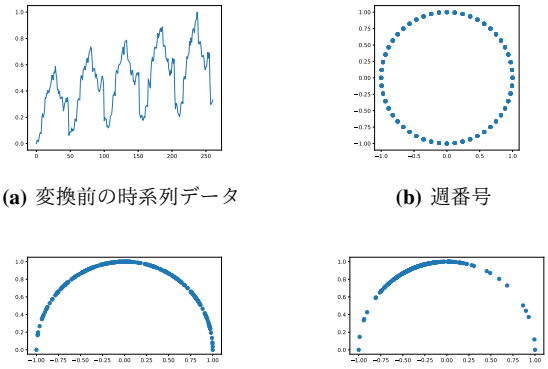
$$dw_t = (\cos(\frac{2\pi w_t}{52}), \sin(\frac{2\pi w_t}{52})) \quad (3)$$

$$d_t = (\cos(\pi x_t), \sin(\pi x_t)) \quad (4)$$

x_t，w_t は t 時点におけるデータの値，d_t，$dw_t \in \mathbb{R}^2$ は時刻 t における拡張後の特徴量を表している．週番号以外について，式 (4) で 2 次元に拡張する際には，事前に各時系列データの最小値が 0，最大値が 1 になるように正規化を行う．図 3a の時系列データに対して，式 (3)，式 (4) で拡張した際の値の分布を図 3b，図 3c，図 3d に示す．図 3b が週番号，図 3c が各時刻における値，図 3d が 1 時刻前との差分である．週番号に関しては，図 3b のように 52 時刻で円状を 1 週するように配置されている．多くの時系列データは 1 年で周期をもつ．さらに 52 週目と 1 週目は連続しているため，隣接させることが効果的だと考え，このような拡張を行った．一方，各時刻における値や差分は，最小の値と最大の値が隣接しないように，0〜π の範囲で拡張した．

3.4. Embedding Layer

この層では，入力されたデータの次元の拡張と位置情報の付与を行う．時系列データの予測において，入力データの次元を拡張せずに用いる場合，その次元は



(a) 変換前の時系列データ　**(b)** 週番号

(c) 各時刻における値　**(d)** 1 時刻前との差分

図 3: 単位円上への拡張

1 であり，ネットワークのノード数も 1 になる．しかしノード数が 1 の場合，学習できる情報量が少ないため，時系列データの性質を学習することが出来ない．[4] では，単語埋め込みをして，次元を 512 に拡張している．そこで，本稿においても，入力データの次元の拡張を行う．まず，入力されたデータを 3.3 節の手法で 6 次元に拡張する．その後，さらに次元を拡張させ，位置情報の付与を行う．Transformer では，再帰や畳み込みなどの処理を行わない．そのため，データに対して Positional Encoding を行い位置情報を埋め込むことで，時間的関係を考慮する．本稿では，次元を拡張した後，Positional Encoding を行う．[4] では，Positional Encoding として，sin 関数，cos 関数を用い，データに周期のある波を与え，相対位置を付与している．本稿では，[4] で用いられていた Positional Encoding のパラメーターを一部変更し，適用した．これらの計算は，入力を x，出力を y としたとき，以下の式で表される．

$$y = max(Emb(x)W_1 + b_1, 0)W_2 + b_2 + PE \tag{5}$$

$$PE_{(t,2i)} = \sin(t/1000^{2i/D})/100 \tag{6}$$

$$PE_{(t,2i+1)} = \cos(t/1000^{2i/D})/100 \tag{7}$$

$Emb(x) \in \mathbb{R}^{N\times T\times 6}$ は入力 $x \in \mathbb{R}^{N\times T}$ を 3.3 節の手法で拡張した値である．また，$W_1 \in \mathbb{R}^{6\times D}$ と $W_2 \in \mathbb{R}^{D\times D}$ はそれぞれの層の重みであり，b_1，b_2 はバイアスである．ここで，N は時系列データの種類数，T は時系列データの長さ，D は拡張後の次元を表している．また，PE は Positional Encoding であり，t は時間次元でのインデックス，i は特徴量次元でのインデックスを表している．i が偶数の時，式 (6) を足し，奇数の時は式 (7) を足す．[4] では，sin，cos の値を除算していないため，Positional Encoding の値域 y_p は $-1 \leq y_p \leq 1$ である．しかし，入力データの最小値 0，最大値 1 に対して Positional Encoding の範囲が $-1 \leq y_p \leq 1$ では，Positional Encoding の値が大きすぎてしまい，入力データの持つ性質が消えてしまうと考えた．そこで本稿では，Positional Encoding の値を 1/100 にした．また，[4] では，sin，cos の中で，t を 10000 で除算している．この 10000 とは，文書中の総単語数を想定したものである．本稿の対象とする時系列データは高々200 データであるため，除数を 1000 にした．

3.5. Time Transformer

2 章で述べたように，Transformer を用いた予測において，他の時系列データを複数束ねて特徴量として用いると，それがノイズとなってしまい，むしろ予測性能が低下してしまう．特に本稿で想定しているデータは，高々2，3 年分のデータであるため，他の時系列データとの関係性を学習しきれず，ノイズとなってしまう．この場合，まずは自身の過去のデータについて，重点的に学習して予測を行い，その後，他の時系列データとの関係性を用いて，自身の予測を補うことが予測性能の向上につながると考えた．そこで本稿では，自身の時系列データのみから学習を行う Time Transformer と，時系列データ間の関係性を学習する Space Transformer を用いることで，それぞれの学習に特化させ，予測性能の向上を目指す．

Time Transformer では，入力されたデータ $X \in \mathbb{R}^{N\times T\times D}$ の時系列方向に関して学習を行う．N は時系列データの系列数，T は得られている時系列データの長さ，D は特徴量の数を表している．時系列方向への学習は，それぞれの時系列データごとに独立して行われる．

Time Transformer では，まず入力されたデータに対して Self-Attention[4] を行い，過去のどの時点と関連があるか計算し，各時点との相互関係の重要度を求める．Time Transformer における Self-Attention では，未来の情報を参照できないようにマスクをかける．

Attention の計算後，出力結果と Attention の計算前の値に対して残差接続を行い，Layer Normalization をする．残差接続では，一つ前の層への入力と，その層の出力を足し，出力する．この構造は，元々画像処理の分野で提案された ResNet というモデルで取り入れられた手法であり，層が深くなっても勾配の消失が起こりにくく，学習が進みやすいという利点がある [9].

その後，Feed Forward Netwark(以降 FFN)[4] で，Attention 層で計算したスコアを用いて，学習を行う．最後に，FFN の出力と，FFN への入力を用いて，残差接続と Layer Normalization を行う．Time Transformer の出力 y_t は $y_t \in \mathbb{R}^{N\times T\times D}$ となる．

3.6. Space Transformer

Time Transformer では個別の時系列データごとに時間方向について学習を行った．Space Transformer ではその値を用いて，それぞれの時刻に対して，その時刻における時系列データ間の関係性を学習させる．

Space Transformer は最初の処理として，Time Transformer の出力 $y_t \in \mathbb{R}^{N\times T\times D}$ の次元を入れ替える．入れ替えた後の値 x_s は $x_s \in \mathbb{R}^{T\times N\times D}$ となる．次に，次元を入れ替えたデータに対して Self-Attention を行う．Space Transformer での Self-Attention は同時刻における時系列データ間の関係性を計算をする．Space Trasformer における Attention では，時刻ごとにそれぞれ独立して，時系列データ間の関連度を計算しているため，未来の情報を参照することがない．そのため，Space Transformer では，マスクは不要である．その後，Attention で計算したスコアに基づいて，FFN で時系列データ間の関係性を学習する．最後に，入れ替えていた次元を，元の次元に戻し，出力する．すなわち，Space Transformer の出力 y_s は $y_s \in \mathbb{R}^{N\times T\times D}$ となる．

3.7. Output Layer

Output Layer では，これより前の層で計算した情報をもとに，予測結果を出力する．Output Layer への入力 x に対して出力 y は以下のように計算される．

$$y = max(xW_1 + b_1, 0)W_2 + b_2 \tag{8}$$

$W_1 \in \mathbb{R}^{D\times D}$,$W_2 \in \mathbb{R}^{D\times 1}$ はそれぞれの層における重みであり，b_1,b_2 はバイアスである．[4] では，出力層に softmax 関数用いているが，時系列データの予測においては，値を出力する．そのため，最終層を線形結合層にすることで，予測値を出力する．

3.8. 学習

データ数が少ない場合，ネットワークは過学習を起こし，テストデータに対する予測性能が低下する．そのため，学習データにノイズを載せることで，学習データのバリエーションを増やし，過学習を抑制した．本稿では 2 種類のノイズを用いる．

1 つめのノイズは，時系列データの値全体を変更するノイズである．このノイズを載せることで，時系列データ全体のスケールを変更する．本稿では，データのスケールを 0.8 倍〜1.2 倍に変更しながら学習を行った．

2 つめのノイズは，時刻ごとに加算するノイズである．このノイズは，各時系列データの各時刻に対して，それぞれ微小な値を加算する．本稿では，各時刻に対してその時刻における値の約 3% のノイズを加減算した．

3.9. 損失関数

学習の目的は，予測値と実績値の差を小さくすることである．そのため，本稿では，学習時の目的関数として，以下の二乗誤差関数を用いた．

$$loss = \frac{1}{2}|y_t - \hat{y}_t|^2 \tag{9}$$

y_t が正解値，$\hat{y}_t$ はモデルの予測値を表している．学習時にはこの $loss$ を最小化するように学習を行う．

4. 実験

4.1. 対象となる実験データ

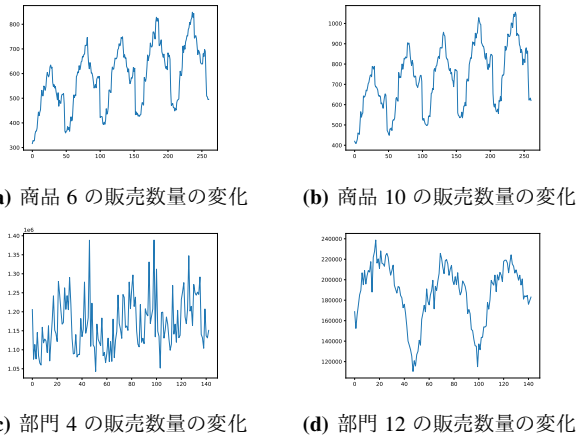
Store Item

このデータは複数のとある店舗における商品ごとの販売数量が日ごとに記録されている [10]．今回の実験では，各店舗がもつ商品の日ごとの販売数量を，店舗は区別せずに，同じ商品，同じ日付をもつデータの販売数量で足し合わせ，その後，各商品の日ごとの販売数量を 7 日間ずつ足し合わせた．結果として，50 商品，261 週間分のデータが記録されている．

Retail Data

このデータはある小売店の各店舗における部門ごとの販売量が週ごとに記録されている [11]．今回の実験では，店舗は無視して，同じ部門，同じ週番号を持つデータの販売数量を足し合わせた．結果として，74 部門，143 週間分のデータが記録されている．

各データセットの一部を図 4 に示す．図の横軸が時間軸，縦軸が販売数量である．今回の実験で用いた 2 つのデータセットはそれぞれ性質が多少異なる．Store Item はかなり明確な周期が存在し，かつ，他の商品も似た傾向を持つ．それに対して Retail Data は，周期はあるものの，Store Item 程明確ではない．また，他の商品との相関をもつデータがいくつか存在している．実験では，データの 90%を学習に用い，残りの 10%をテストデータに用いた．



(a) 商品 6 の販売数量の変化　**(b)** 商品 10 の販売数量の変化

(c) 部門 4 の販売数量の変化　**(d)** 部門 12 の販売数量の変化

図 4: データの一部

4.2. 評価指標

モデルの性能を評価するための指標について述べる．本稿では評価指標として，sMAPE(Symmetric Mean Absolute Percent Error)，CORR(Correlation Coefficient)，LCORR(Lagged Correlation Coefficient) を用いた．それぞれの評価指標は以下の式で計算される．

$$sMAPE = \frac{1}{T}\sum_{t=1}^{T}\frac{|\hat{y}_t - y_t|}{(|\hat{y}_t| + |y_t|)/2}$$

$$CORR = \frac{\frac{1}{T}\sum_{t=1}^{T}(y_t - \bar{y})(\hat{y}_t - \bar{\hat{y}})}{\sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - \bar{y})^2}\sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{y}_t - \bar{\hat{y}})^2}}$$

$$LCORR = \frac{\frac{1}{T-1}\sum_{t=2}^{T}(y_t - \bar{y})(\hat{y}_{t-1} - \bar{\hat{y}})}{\sqrt{\frac{1}{T-1}\sum_{t=2}^{T}(y_t - \bar{y})^2}\sqrt{\frac{1}{T-1}\sum_{t=2}^{T}(\hat{y}_{t-1} - \bar{\hat{y}})^2}}$$

y_t は，時系列データの t 時点における実測値，$\hat{y}_t$ は予測モデルが出力した時系列データの t 時点における予測値，T はテストデータの数を示す．sMAPE は正解値と予測値の差を，0 以上 2 以下の相対誤差で表す．sMAPE は小さいほうが良いと言える．

CORR は正解値と予測値の相関係数を表しており，LCORR は予測値を 1 時刻遅らせた時の正解値との相関係数を表している．モデルが予測値として 1 時刻前の正解値に近い値を出力するとき，CORR に比べて，LCORR は大きくなる．CORR が高く，LCORR が CORR に比べて小さいと，単に 1 時刻前の値を用いて予測しているのではなく，過去の傾向から予測を行えているといえる．

4.3. 比較対象のモデル

ARIMA

パラメーターは AIC(赤池情報量) を基準に，grid search を用いて求めた．AIC はモデル選択のための指標であり，データのモデルへの当てはまり具合と，パラメータの数で評価する．

LSTM

このモデルは時間的依存関係を学習するための LSTM 層が 1 層，結果を出力するための全結合層 1 層からなる．LSTM 層の Unit 数は 128, 出力層の次元は 1 である．

Simple

このモデルは提案手法の中の，Embedding Layer, Time Transformer，Output Layer だけで構成されたモデ

ルである．しかし，ここで用いる Embedding Layer は，特徴量を極座標系の値に変換しない．すなわち，使用する特徴量は絶対値としての意味を持った値を使用する．また，値をそのまま使用するため，Time Transformer 内部で Layer Normalization を行わない．

Simple(Theta)

このモデルは Simple と同様の構成である．Simple との違いとして，Embedding Layer 内で，入力特徴量を極座標系の値に変換する．また，極座標系の値に変換を行っているため，Layer Normalization を使用する．

Simple(多商品)

このモデルは Simple(Theta) と同様の構造である．異なる点として，入力データに自身のデータだけでなく，他の時系列データも束ねて用いた．

提案手法

このモデルは Time Transformer と Space Transformer を交互に用い，最後に Time Transformer を使用したモデルである．今回の実験では，Time Transformer と Space Transformer を 2 回繰り返した後に，Time Transformer を 1 回用いた．

4.4. 実験結果

4.1 で述べたデータセットに対して，予測時点の 1 時点前までの実際の値を用いて，予測を行った．すなわち，テストデータの h 番目の値を予測する際の，入力 x は以下のようになる．

$$x = (y_1^1, y_2^1, ..., y_{T+h-1}^1, y_1^2, ..., y_{T+h-1}^2, ..., y_{T+h-1}^n) \quad (10)$$

ここで，y_t^i は i 個目の時系列データの t 時点の値を表している．また，時点 T までが，学習に用いた値である．ニューラルネットワークを用いたモデルは，重みとバイアスの初期値による学習結果のばらつきを抑えるために，それぞれのモデルで 10 回実験を行った．ARIMA モデルに関しては，初期値によるばらつきは起こらないため，1 回のみ実験を行った．

各データセットに対して，各モデルで 10 回予測を行った時の各評価指標の平均値と標準偏差を表 1 に示す．実験の結果，Store Item に対する予測では，提案手法が sMAPE，CORR ともに最も良い結果となった．また，LCORR の値も，CORR と比較して小さくなっている．これは，単に 1 時刻前の値を参考に出力しているのではなく，過去の傾向全体から，予測を行えていることを示している．一方で，Retail Data に対しては，sMAPE では LSTM が，CORR では提案手法が最も良い結果となった．どちらのモデルも LCORR は CORR より小さく，単に 1 時刻前の値を出力しているだけではないことを示している．

Simple と Simple(Theta) を比較すると，どちらのデータセットに対する予測でも，入力に使用する情報を極座標系の値に変換して用いたほうが性能が良かった．

特徴量として，他商品のデータを用いた Simple(多商品) は，どちらのデータセットに対しても，予測性能が低下しており，他商品の情報がノイズになってしまうことが示された．

一方，Simple(Theta) と提案手法を比較すると，どちらのデータセットに対しても，Space Transformer を用いた提案手法の方が sMAPE，CORR ともに改善している．LCORR の値も CORR に比べて減少していることから，Space Transformer 導入することで他商品の情報がノイズとならず，むしろ予測に貢献できるようになったことが示された．

表 1: 直前までの実績を用いた予測結果

Dataset		Store Item		Retail Data	
モデル	評価値	平均	標準偏差	平均	標準偏差
ARIMA	sMAPE	0.063	-	0.132	-
	CORR	0.884	-	0.307	-
	LCORR	0.928	-	0.489	-
LSTM	sMAPE	0.053	0.029	**0.089**	0.005
	CORR	0.943	0.064	0.654	0.017
	LCORR	0.896	0.041	0.609	0.020
Simple	sMAPE	0.075	0.022	0.189	0.054
	CORR	0.857	0.033	0.333	0.099
	LCORR	0.821	0.028	0.408	0.144
Simple(Theta)	sMAPE	0.028	0.006	0.111	0.008
	CORR	0.986	0.002	0.581	0.023
	LCORR	0.918	0.008	0.369	0.041
Simple(多商品)	sMAPE	0.063	0.014	0.229	0.034
	CORR	0.925	0.009	0.559	0.037
	LCORR	0.958	0.005	0.383	0.037
提案手法	sMAPE	**0.024**	0.002	0.099	0.013
	CORR	**0.987**	0.002	**0.675**	0.038
	LCORR	0.930	0.012	0.517	0.046

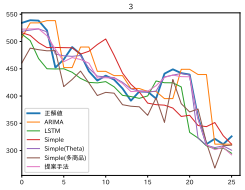


図 5: 部門 3 の予測結果

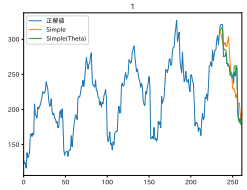


図 6: 商品 1 の予測結果

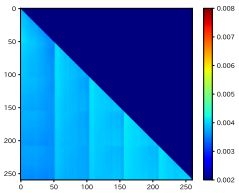


図 7: Simple の Attention Score

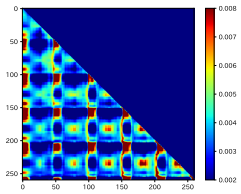


図 8: Simple(Theta) の Attention Score

図 5 に Retail Data の部門 3 に対する各モデルの予測結果のグラフを示す．グラフの縦軸が販売量，横軸が予測をした週である．また，図中の青線が正解値であり，その他の色が各モデルの予測値を表している．グラフを見ると，15 週目付近の販売量が増加する箇所と，20 週目付近の販売量が減少する箇所に対して，橙色で表した ARIMA モデルによる予測は，それぞれの変化に対して遅れて予測を行っていることが確認できる．一方，桃色で表した提案手法による予測や，緑色で表した LSTM による予測は，それらの変化の起きる時期を予測できていることが確認できる．

5. 考察

Transformer による予測において，入力データを極座標系で表現することで，予測性能が向上している．これは，[4] で提案された Transformer は入力に単語のベクトルを想定しているからだと考える．まず，Self Attention は内積を用いるため，cos 類似度に基づいてデータの相互関係の重要度を計算している．そのためデータは絶対値としてではなく，ベクトルとして意味を持つ必要がある．そのため，値をそのまま用いるよりも Attention が有効になったのではないかと考える．Layer Normalization についても同様で，正規化の際に絶対値の持つ意味は

消えてしまう．そのため，絶対値として意味を持たせるのではなく，位相として意味を持たせる必要があったと考える．Store Item の商品 1 に対して，値をそのまま絶対値として用いた Simple と，値を位相として用いた Simple(Theta) で予測を行った際の，Attention スコアを図 7 と図 8 に示す．図 6 は，Store Item の商品 1 に対する Simple と Simple(Theta) の予測結果を表した図であり，図中の青線が，学習部分を含めた商品 1 の実際の値，橙線が Simple の予測結果，緑線が Simple(Theta) の予測結果を示している．そして，図 7 と図 8 は，Simple と Simple(Theta) の Time Transformer 内で計算した Self Attention の結果をヒートマップにしたものである．図の縦軸と横軸がそれぞれ時刻であり，図内の色が，縦軸と横軸が交差する時刻における関係性を計算した値を表している．青に近いほど，Attention のスコアが低い，すなわち関係性が低く，赤に近いほど Attention のスコアが高く，関係性が高いことを示している．また，図中の右上に関しては，未来の情報であるため，マスクがかかっており，青色になっている．図を見ると，Simple(Theta) では，格子状に Attention Score の大きいところと小さいところが存在している．商品 1 の販売量が小さくなる 200 週付近の予測を行う際に，200，150，100，50，0 週付近の Attention Score が大きくなっている．商品 1 は，約 50 週で周期があるため，予測する週の直近の値と，1 周期前，2 周期前，3 周期前，4 周期前の値を参照している．また，販売量がピークを迎える 225 週付近の予測では，225，175，125，75 週付近の Attention Score が大きくなっている．これについても同様に，直近の値と 1 周期前，2 周期前，3 周期前の値を参照していることが確認できる．1 周期前の値は，予測時点の値と傾向が似ているため，モデルはデータの性質を学習できていると言える．

一方で Simple の Attentoin Score を見ると，若干の縞模様が見えるものの，どの時点においても Score がほとんど変化していない．50，100，150，200 週を少し過ぎた時点において，多少他の時点よりも青の色が薄く，Attention Score が大きくなっている．これは，商品 1 の販売量が小さくなる時期と一致している．すなわち，モデルはどの時点の予測においても販売量が小さくなった時の値を他の値より若干重視して，予測している．販売量の大きい時期については，過去の販売量の大きい時期を参照することが，重要だと考えているため，モデルが時系列データの性質を十分に学習できているとは言えない．しかし，販売量の小さくなる週の傾向が似ていることはわずかに学習できている．この理由として，入力データを絶対値としてそのまま学習に利用した場合においても，ノード数を拡大した際に，ニューラルネットワークが値から位相情報をわずかに学習したのではないかと考えた．しかし，入力データを Embedding Layer で位相に変換することで，データをより正確に位相情報に変換できるため，Simple(Theta) の方がデータの性質を学習でき，予測性能が向上したのではないかと考える．

Time Transformer のみを用いたモデルよりも，Space Transformer を用いながら予測を行うモデルの方が性能が良かった．データ数が少ない場合，自身の時系列データのみだと，学習を十分に行えない．これは，学習データにノイズをのせて，データのバリエーションを増やしたとしても，その時系列データが本来持つ性質自体は変わらない．そのため，Space Transforme によって時系列データ相互で予測値を補完しあうことが有効だったのではないかと考えた．

最後に，ARIMA モデル，LSTM モデルと，提案手法の比較を行う．ARIMA モデルは予測の際に，常に同じ時刻分前の値を参照する．実際に，ARIMA モデルの予測は，すべて場合で CORR よりも LCORR のほうが高い．ARIMA モデルが長期の時間関係をとらえていないからであり，昨年と比較してピークが早まりそうということは予測できない．一方で LSTM や Transformer では，その時刻に応じて，過去のどの時点を参照すればよいか変えられるため，予測をより柔軟に行うことができる．LSTM は，入力を逐次的に処理するため，過去の値を直接参照することが出来ない．一方，提案手法では時間情報について，Self Attention を用いて，並列に処理を行うため，過去の値をそのまま残すことが可能である．そのため，Store Item のような周期が明確なデータに対しては，性能高く予測することが可能になったのではないかと考えた．実際に，図 8 では 1 周期，2 周期前の Attention スコアが高くなっており，その値を予測に用いることが出来ている．

6. むすび

本稿では Transformer をベースにした予測モデルの構築を行った．入力データから生成した特徴量を，極座標系の 2 次元に拡張し，方向で表現することで，値を絶対値としてそのまま用いるよりも予測性能が向上することを示した．また，時系列方向の学習を行う Time Transformer と時系列データ間の関係性を学習する Space Transformer を組み合わせることで，時間方向に関するデータが少ないデータセットにおいて，Time Transformer のみを用いるよりも予測性能が向上することを示した．

Positional Encoding やモデルのノード数の最適なパラメーター探索と，より汎用性の上がるノイズの載せ方については今後の課題とする．

参考文献

[1] Michael Chui, James Manyika, Mehdi Miremadi, Nicolaus Henke, Rita Chung, Pieter Nel, and Sankalp Malhotra. Notes from the AI frontier: Applications and value of deep learning. *Discussion Paper*, 2018.

[2] Box. G.P.E and G.M. Jenkins. *Time Series Analysis ; Forecasting and Control*. Holden-Day, 1970.

[3] Sepp Hochreiter and Jugen Schmidhuber. Long short-term memory. *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[5] 本田良司, 小笠原範光, 小玉亮. Transformer による時系列予測. 人工知能学会全国大会論文集, 2020.

[6] Neo Wu, Bradley Green, and Shawn O'Banion. Deep transformer models for time series forecasting: The influenza. *arXiv:2001.08317*, 2020.

[7] Mingxing Xu, Wenrui Dai, Chunmiao Liu, Xing Gao, Weiyao Lin, Guo-Jun Qi, and Hongkai Xiong. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv:2001.02908*, 2021.

[8] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *Neural Information Processing Systems - Deep Learning Symposium*, 2016.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[10] Kaggle. Store item demand forecasting challenge. https://www.kaggle.com/c/demand-forecasting-kernels-only/data(参照 2022-02-03).

[11] Manjeet Singh. Retail data analytics. https://www.kaggle.com/manjeetsingh/retaildataset(参照 2022-02-03).